IMPLEMENTATION OF COSINE MODULATED DIGITAL FILTER BANK ON PROCESSOR WITH ARM ARCHITECTURE

DOI: 10.36724/2072-8735-2020-14-11-57-63

Kirill Yu. Sokolov,
Moscow Technical University of Communications and Informatics (MTUCI), Moscow, Russia, ***sokolovkirilluy@gmail.com***

Vladimir S. Priputin,
Moscow Technical University of Communications and Informatics (MTUCI), Moscow, Russia, ***v.s.priputin@mtuci.ru***

Elizaveta O. Lobova,
Moscow Technical University of Communications and Informatics (MTUCI), Moscow, Russia, ***lizabeth2@mail.ru***

Manuscript received 14 September 2020;
Accepted 20 October 2020

Keywords: *cosine modulated filter bank, discrete cosine transform type 4, graphics processing unit, OpenCL, ARM processors, computational complexity*

This paper presents a class of multichannel cosine-modulated filter banks (CMFB) of analysis based on the modulation effect with a fast discrete-cosine transformation of the fourth type (DCT-IV), which is calculated using the fast Fourier transform. As a prototype filter, a low-frequency filter with a finite pulse characteristic was used, frequency-shifted copies of which were made using an effective technology for polyphase representation of the filter Bank. The comparison of the number of arithmetic operations performed by digital down converter (DDC) based on cascade integral-comb (CIC) and CMFB based on different number of channels is given. A software description of the CMFB algorithm is presented in the form of block diagrams describing the capabilities of the Opencl and clfft software libraries for implementing the DCT-IV filter Bank and modulation algorithm on a GPU. The obtained algorithm was tested on an ARM family processor and a mali GPU with a table with sample rate for different number of channels with the maximum load of the graphics processor (GPU) and the minimum load of the Central processor (CPU).

Information about authors:
Kirill Yu. Sokolov, *Moscow Technical University of Communications and Informatics (MTUCI), engineer of the first category NIL-4807 NICH MTUCI, Moscow, Russia*
Vladimir S. Priputin, *Moscow Technical University of Communications and Informatics (MTUCI), Ph.D. head of laboratory NIL-4807 NICH MTUCI, Moscow, Russia*
Elizaveta O. Lobova, *Moscow Technical University of Communications and Informatics (MTUCI), junior researcher NIL-4803 NICH MTUCI, Moscow, Russia*

Для цитирования:
Соколов К.Ю., Припутин В.С., Лобова Е.О. Реализация косинусно-модулированных цифровых фильтр банков на базе процессора с архитектурой ARM // T-Comm: Телекоммуникации и транспорт. 2020. Том 14. №11. С. 57-63.

For citation:
Sokolov K.Yu., Priputin V.S., Lobova E.O. (2020) Implementation of Cosine Modulated Digital Filter Bank on Processor with ARM architecture. *T-Comm*, vol. 14, no.11, pp. 57-63. (*in Russian*)

Introduction

One of the most effective tools for building broadband data transmission systems is the digital filter bank, which is a set of similar digital bandpass filters designed to divide the input signal samples into several sub-bands (sub channels) that evenly cover the entire frequency range under study.

Figure 1 shows a typical scheme for building M-channel filter banks. At the first stage, the input samples of the signal fall into the delay line, and the signal is split into sub-band components using analysis filter banks. Since the received sub-band signals have a limited frequency band, it is necessary to lower their sampling rate, which is carried out using decimators, and the decimation coefficient depends on the number of channels. At the second stage, additional processing is performed in the sub-band processing unit in order to clear the noise. At the third stage, the signal is restored from the sub-band components using the analysis filter banks, where the sampling rate is reversed by interpolating the signal samples. Obviously, the use of decimators and interpolators in sub-band processing creates the problem of decimation noise at the analysis stage and interpolation errors at the synthesis stage, but reduces the number of mathematical operations.

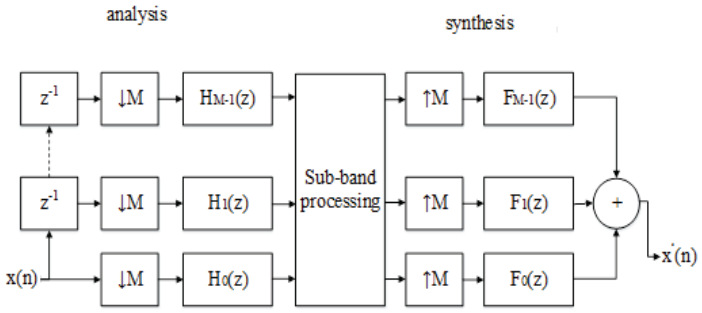

Fig. 1. Basic block diagram of filter banks representation

Filter banks based on the modulation effect and the use of a low-pass filter of the prototype are widely used, followed by obtaining its frequency-shifted copies so that the set of filters covers the entire operating frequency range.

Cosine modulated filter bank

Cosine-modulated filter banks do not always have full recovery properties, but by optimizing the coefficients of the prototype filter, you can achieve the required accuracy. However, since only the analysis filter bank is considered in this article, the expression for the impulse characteristics of filters takes the following form:

$$h_p(n)=\sqrt{\frac{2}{M}}h(n)\cos\left(\left(k+\frac{1}{2}\right)\frac{\pi}{M}\left(n+\frac{M+1}{2}\right)\right) \quad (1)$$

M – the number of channels, $k=0,1\ldots M-1$ is the channel number, $n=0,1\ldots 2mM-1, m$ is the overlap coefficient, and $h(n)$ is the coefficients of the prototype filter.

When comparing the modulation sequences and the basis functions of the discrete cosine transformation of the fourth type (DCT-IV), we can conclude that the necessary modulation functions can be implemented using $M*M$ DCT-IV matrices, which are symmetric in structure and satisfy the following rules: $C^TC=CC^T=I$, I is the identity matrix. Modeling has confirmed the effectiveness of this method in constructing digital filter banks with a large number of channels and high frequency selectivity. The amplitude-frequency response of the prototype filter and the resulting cosine-modulated filter bank is shown in figure 2.

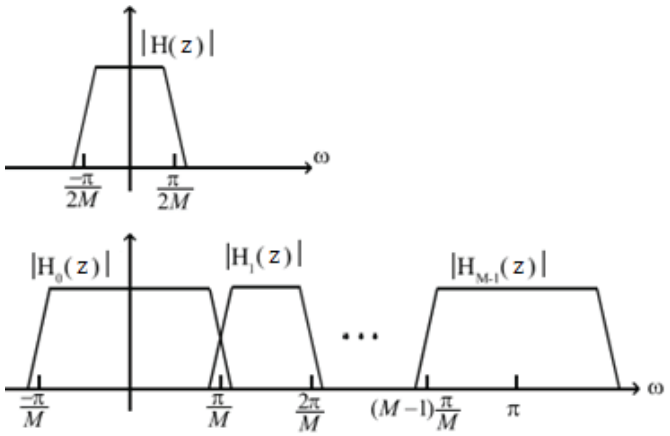

Fig. 2. Frequency response of the prototype filter and the cosine-modulated filter Bank

Polyphase representation of filter banks

The implementation of cosine-modulated filter banks proposed in this article is based on an effective scheme for the polyphase representation of the prototype filter [1] shown in figure 3.

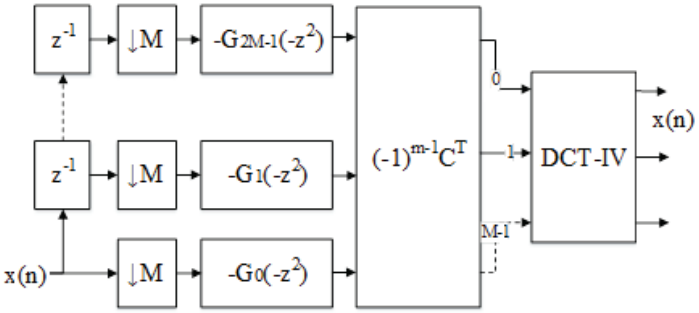

Fig. 3. Block diagram of a cosine modulated filter bank

The algorithm of operation of this filter Bank implies the passage of input samples of the signal through the delay line, where the number of, the number of z^{-1} is equal to $2M-1$, followed by their decimation in ↓M-decimators [7], the number of which is equal to twice the number of channels in the prototype filter (2M), and the decimation coefficient is equal to the number of channels (M).

At the next stage, filtering is performed due to the polyphase representation of the prototype filter, as a result of which the impulse response of the original filter is decomposed as follows:

$$\begin{bmatrix} -G_{2M-1}\left(-z^2\right) \\ -G_2\left(-z^2\right) \\ -G_1\left(-z^2\right) \\ -G_0\left(-z^2\right) \end{bmatrix}, \quad (2)$$

$G_l(\cdot)$ is expressed in the form $2M$ of polyphase components using polyphase filters of type one, $h_p(n)$ is the prototype filter:

$$H_p(z)=\sum_{l=0}^{2M-1}\sum_{p=0}^{m-1}h_p(l+2pM)z^{-(l+2pM)}=\sum_{l=0}^{2M-1}z^{-l}G_l\left(z^{2M}\right) \quad (3)$$

The third stage is the representation of a transposed matrix C containing submatrices f $2M * M$ dimension $M/2 * M/2$, the matrix J is a reverse matrix I:

$$C=\begin{bmatrix} 0 & -I \\ 0 & J \\ J & 0 \\ I & 0 \end{bmatrix}_{2M*M} \quad (4)$$

$$I=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{\frac{M}{2}*\frac{M}{2}} \quad (5)$$

$$J=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_{\frac{M}{2}*\frac{M}{2}} \quad (6)$$

At the end, the calculated values fall into the DCT-IV transformation block, which contains M inputs and outputs.

Fast DCT-IV using fast Fourier transform

At the first stage there is an input vector with real numbers of length M: $x(n)=0,1\ldots M-1$ which is re-formed as follows :

$$x_n \leftarrow x_{2n} \quad (7)$$

$$x_{n+M/2} \leftarrow x_{M-1-2n} \quad (8)$$

$n=0,1,\ldots M/2-1$, the $\leftarrow$ sign indicates the transfer of samples from one part of the vector to another.

At the second stage, need to get an M-element complex vector, which will be divided into two vectors. The first vector will contain real components:

$$x_n \leftarrow \mathrm{Re}\left(\left(x_n+jx_{n+\frac{M}{2}}\right)\exp\left(-j\left(n+\frac{1}{4}\right)\frac{\pi}{M}\right)\right) \quad (9)$$

And the second vector will contain imaginary components:

$$x_{n+\frac{M}{2}} \leftarrow \mathrm{Im}\left(\left(x_n+jx_{n+\frac{M}{2}}\right)\exp\left(-j\left(n+\frac{1}{4}\right)\frac{\pi}{M}\right)\right) \quad (10)$$

Both vectors are added to one vector so that the real part lies in the range $\left[x_0\ldots x_{\frac{M}{2}-1}\right]$, and the imaginary part in the range $\left[x_{\frac{M}{2}}\ldots x_{M-1}\right]$.

In the third step, the fast Fourier transform (FFT) is calculated, in which the first part of the output vector will contain the real part, and the second part of the vector will contain the imaginary part:

$$\left[x_0\ldots x_{M-1}\right] \leftarrow FFT\left\{\left[x_0\ldots x_{M-1}\right],\frac{M}{2}\right\} \quad (11)$$

The fourth step is the calculation of the new coefficients and rewrite them according to the rule:

$$x_n=x_n\exp\left(-j\frac{n\pi}{M}\right) \quad (12)$$

$n=1,2\ldots M/4-1$, with the skip of one coefficient and $M/4+1,\ \ldots, M/2-1$

$$x_{M/4}=\sqrt{2}(1-j)x_{M/4} \quad (13)$$

$$x_{2n} \leftarrow x_n \quad (14)$$

$$x_{M-1-2n} \leftarrow -x_{n+M/2} \quad (15)$$

$n=0,\ 1\ldots M/2-1$. After the product of all calculations and permutations, the output real vector containing the coefficients of the DCT-IV transformation will be obtained, which will be multiplied with the transposed matrix C figure 3 [2].

Computational complexity of cosine modulated filter banks

The computational complexity of the DCT-IV algorithm includes one M/2 complex fast Fourier transform, M/2 multiplications in the second step, as well as (M/2-2) complex multiplications and one multiplication $\sqrt{2}(1-j)$ described in the fourth step, resulting in the complexity of the DCT-IV algorithm:

$$\mu(M)=\frac{M}{2}\log_2 M+M \quad (16)$$

$$\alpha(M)=\frac{3M}{2}\log_2 M \quad (17)$$

And the computational complexity of M-channel cosine modulated filter banks when using a polyphase structure that contains m multiplications and m-1 additions, as well as a mapping matrix that includes M additions, is calculated as follows:

$$\mu(M)=\frac{M}{2}(4m+\log_2 M+2) \quad (18)$$

$$\alpha(M)=\frac{M}{2}(4m+3\log_2 M-2) \quad (19)$$

m – filter overlay coefficient.

Comparison of the number of arithmetic operations performed by CIC-based DDC and CMFB-based filter Bank

The CIC-based DDC consists of cascades, each of which in turn contains a third-order CIC LPF (three integrating and differentiating links each), half-band FIR filters, and one terminal LPF FIR (total filters without CIC LPF length N_i^{fir}, $i = 0 \div K$).Each half-band filter reduces the sample rate by a factor of 2.

The total number of arithmetic operations performed by the CIC-based DDC during the sampling period is:

$$Q_{CIC} = \left(Ad_{ddc} + Mul_{ddc}\right)2M \quad (20)$$

$$Q_{CIC} = 2M\left(2 + 2\left[6 + \sum_{i=0}^{K}\frac{N_i^{fir}-1}{M_{CIC}\cdot 2^i} + \sum_{i=0}^{K}\frac{N_i^{fir}}{M_{CIC}\cdot 2^i}\right]\right) \quad (21)$$

$Ad_{ddc} = 2\left[6 + \sum_{i=0}^{K}\frac{N_i^{fir}-1}{M_{CIC}\cdot 2^i}\right]$ – is the total number of additions in one cascade during the sampling pe od, $Mul_{ddc} = 2 + 2\sum_{i=0}^{K}\frac{N_i^{fir}}{M_{CIC}\cdot 2^i}$ – is the total number of multiplications in one stage during the sampling period, $M_{CIC} = \left\lfloor \frac{2F_s\Delta f_{CIC}}{\Delta f} \right\rfloor$ – decimation coefficient of the CIC filter, $\Delta f = \frac{F_s}{1.5M}$ – gnal band at the DDC output, Δf_{CIC} – CIC filter bandwidth, F_s – sampling frequency, $\lfloor \cdot \rfloor$ – rounding operator to the nearest smallest value.

The filter Bank accepts M samples of the signal at frequency F_s as input, and after the necessary processing, the output contains samples of M-channels analysis. The number of arithmetic operations performed during one sampling period is equal to:

$$Q_{Bank} = Mult_b + Add_b \quad (22)$$

Total number of multiplications per sampling period:

$$Mult_b = \left(\frac{M}{2}\log_2 M + M + 2Mm + m\right)\frac{1}{M} \quad (23)$$

The total number of additions per sampling period:

$$Add_b = \left(\frac{3M}{2}\log_2 M + 2M(m-1) + M\right)\frac{1}{M} + (2m-1)\frac{1}{2M} \quad (24)$$

The dependence of the number of arithmetic operations on $2M$, $M = 2^r$, $r = 1,...10$ is shown in figure 6. Calculation was performed at $F_s = 12$ MHz. The following conclusions can be drawn from these dependencies:

1) The number of computational operations increases with the increase of *2M*;

2) DDC based on the filter Bank requires a minimum of 6.4 times at *2M*=4 and a maximum of 369.3 times at *2M*=2048 fewer arithmetic operations than DDC based on CIC.

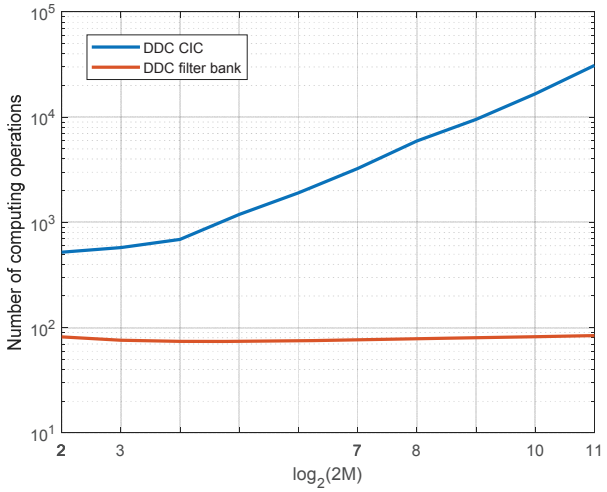

Fig. 4. Dependence of the number of arithmetic operations on 2M

Program implementation of cosine modulated filter banks

In this article, the cosine modulated filter Bank is implemented on an arm processor rk3399pro with an integrated GPU mali mp4 860-t, the algorithm itself is implemented using the programming languages C/C++ and OpenCL, mathematical operations of the FFT, DCT-IV algorithms and working with matrices are performed on the GPU.

To implement the fast Fourier transform algorithm, it was necessary to programmatically include the clfft and OpenCL libraries, as well as install video card drivers. OpenCL is an open standard for low-level programming, which is used by such IT product manufacturers as AMD, ARM, NVidia, Intel, etc. [3]. It allows you to use the processing power of the GPU. Clfft is an additional library containing FFT functions [4].

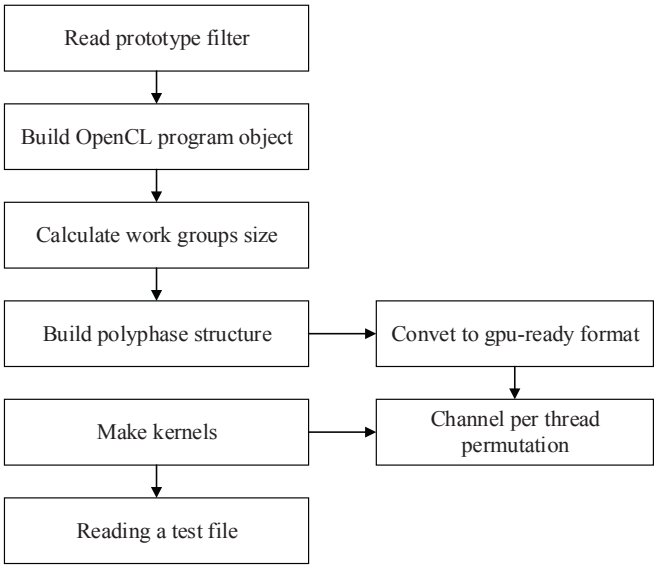

Fig. 5. OpenCL preparatory stage

The whole organization of the project can be divided into the preparatory and processing phases.

For the first one, you need to have records of the broadband signal, the generated pulse characteristic of the prototype filter, and to configure the environment in which the calculations will be performed, you need to perform such manipulations as shown in the block diagram figure 5.

The first block reads a file with a prototype filter that is generated using the "MATLAB" algorithm. if the reading was successful, the second block builds an OpenCL object: it creates a context, device, and Program, which is passed the path to the resource file with the function to run on the GPU. After successful creation of all objects, it is necessary for the algorithm to work effectively to determine the size of the working group W, which will correspond to the number of channels in the filter bank, if $2M > W_{max}$, W_{max} is the maximum size of the working group, then $W = 2 * \frac{M}{W_{max}}$, otherwise $W = 4$, $W_{max}=M/2$. In the block for creating a polyphase structure, the polyphase components are prepared and sorted, which is described by formulas 2 and 3, and is also expressed as an algorithm in figure 6.

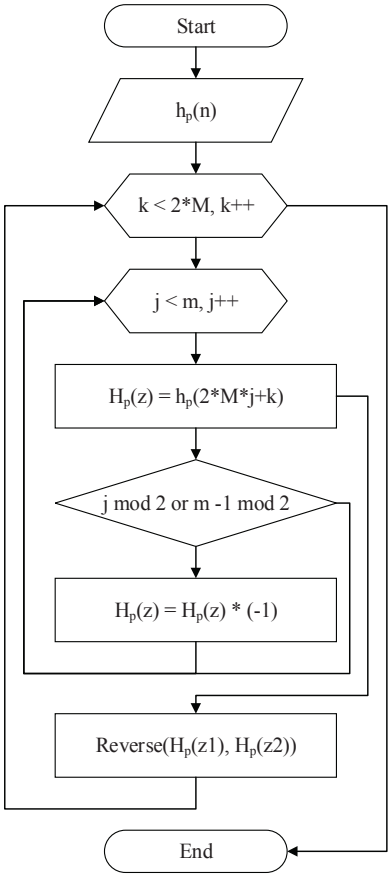

Fig. 6. Building a polyphase structure

Where k – polyphase number count, j-polyphase impulse count, the reverse function - reverses the order of elements in the interval, starting with one element and ending with the second, the coefficients of the polyphase structure are expressed by the following formulas:

$$z = (2 * M - 1 - k) * m + j, \quad (25)$$

$$z1 = (2 * M - 1 - k) * m, \quad (26)$$

$$z2 = (2 * M - 1 - k) * m + m, \quad (27)$$

After these manipulations, a kernel "Channel per thread permutation" is created ready to run on the GPU, which performs a permutation of polyphase components as in matrices 4, 5, 6 for the DCT-IV transformation. It is also necessary to transmit the read test file with a broadband signal recording directly during calculations. It is worth noting that the algorithm can be configured to work in real time.

At the preparatory stage, it was necessary to configure the clFFT environment for fast Fourier transform on the GPU figure 7.

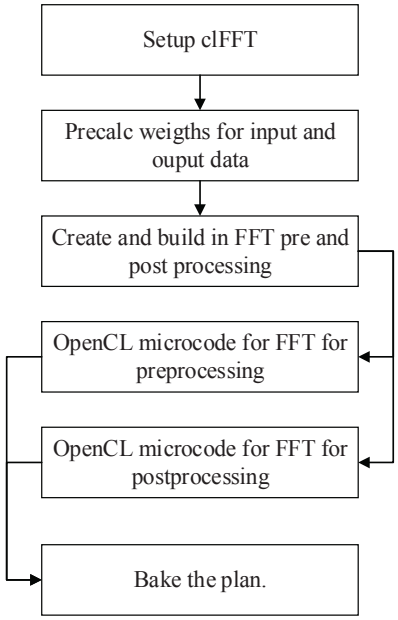

Fig. 7. DCT-IV calculation

The first block initializes the clFFT API, which allows the library to create the resources needed to manage the plans that will be created and destroyed. Also in this block, settings are made such as: setting the accuracy of floating-point FFT data, determining the expected location of input and output buffers, and setting the number of discrete arrays that the plan can process simultaneously. The clfft library supports FFT up to three dimensions (1D, 2D, 3D). This article uses a one-dimensional complex DFT:

$$\bar{x}_j = \frac{1}{scale} \sum_{k=0}^{n-1} x_k \exp\left(\pm i \frac{2\pi jk}{n}\right) \text{ for } j = 0, 1, \ldots, n-1 \quad (28)$$

In the second block, the weights for input and output data are pre-calculated using formulas 7 and 8. the Third block configures and creates pre – and post-processing functions that make it possible to call the built-in OpenCL functions provided by the user for subsequent processing of output data from the FFT core, which significantly improves system performance [5]. "Bake the plan" means compiling the FFT plan.

Individual threads and parts of the code are managed using events, which organize a queue of executed commands (figure 8). To implement the algorithm in this article, you will need 4 events: wait for buffer shifting, wait for buffer reading from GPU, wait for buffer write to GPU, and wait data writing for kernel execution. The second block represents a loop in which all operations will be performed in turn until the results are read from the GPU buffer.

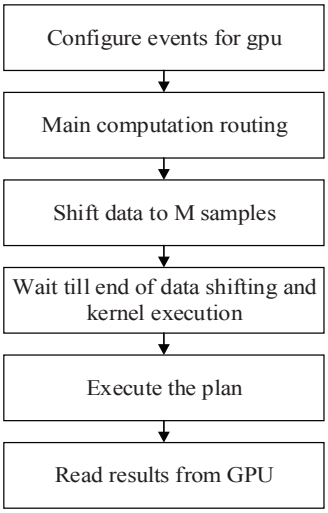

Fig. 8. Main cycle for calculating the filter bank

Testing the implemented algorithm on an ARM processor

The resulting algorithm was tested on ARM RK3399pro [9], which is a six-core processor of the big. LITTLE architecture with 2x Cortex A72 cores up to 1.8 / 2.0 GHz, 4x Cortex A53 cores 1.4 GHz, and an Arm Mali-T860 MP4 GPU with OpenCL 1.2 and DX 11 support, and an NPU up to 3 TOPS [6], most of the mathematical operations are performed on the GPU. The algorithm (figure 5) selected the size of the working group W = 16, and the number of counts per core equal to 8, which with these parameters gave the maximum GPU load and the minimum CPU load, which was 3-6 percent.

Prototype filter type	M	m	Filter FIR length	Average sample rate, ksample/s	Single channel sample rate, ksample/s
ILS_M16_m16	16	16	512	15935	995.97
ILS_M32_m16	32	16	1024	15754.6	492.33
ILS_M64_m16	64	16	2048	15126.4	236.35
ILS_M128_m16	128	16	4096	15041.6	117.51
ILS_M256_m16	256	16	8192	15800	61.7

Fig. 9. Results of testing on the mali-t860 GPU

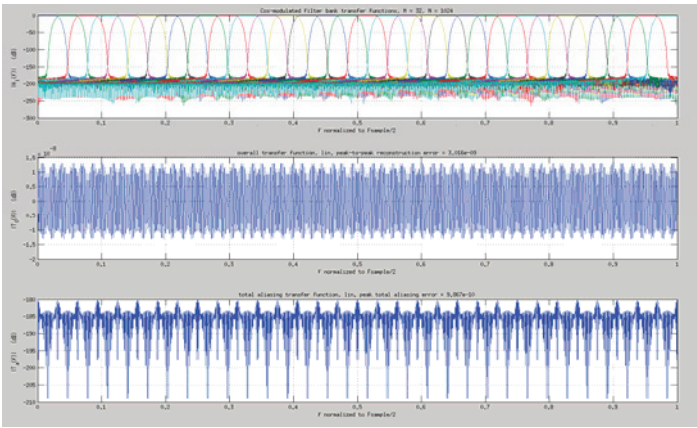

Fig. 10. Filter Bank for 32 real / 64 complex channels

Figure 9 shows the results of testing the implemented algorithm on the mali-t860 GPU, M16-M256 is the number of channels, m16 is the overlap coefficient, and the number of processed blocks is 32768 per cycle, with a frequency of polling the speed of operation once every 40 cycles. It is worth noting the direct proportional dependence of the sample rate per channel on the total number of channels, which indicates that there are no performance leaks in the algorithm.

Figures 10 show such characteristics as transfer function, overall transfer function and total aliasing transfer function for 32 real / 64 complex channels.

Conclusion

In this article, a theoretical description of CMFB was demonstrated with a representation of the calculation formulas and flowcharts of the algorithm. The evaluation of computational costs performed by DDC based on CIC and CMFB based filter Bank showed that DDC CMFB requires at least 6. 4 times at 2M=4 and a maximum of 369.3 times at 2M=2048 fewer arithmetic operations than DDC based on CIC. When testing the algorithm on an ARM processor with an integrated graphics processor, it was found that the proposed algorithm uses the GPU to the maximum and the minimum by 3-6 percent of the CPU with a different number of channels, which will be effectively combined with the algorithm described in the article [8], in which the demodulator channels were filtered on the CPU. A direct proportional dependence of the sample rate of one channel on the total number of channels is revealed. Graphs of filter banks for 32 real / 64 complex channels are constructed.

References

1. Ari Vi lainen, Juuso Alhava, and Markku Renfors, “Efficient Implementation of Complex Modulated Filter Banks Using Cosine and Sine Modulated Filter Banks”, *Hindawi Publishing Corporation EURASIP Journal on Applied Signal Processing* Volume (2006), Article ID 58564, pp. 1-10.

2. Henri e S. Malvar (1992), “Signal Processing with Lapped Transforms”, Artech House, INC. pp 67-75.

3. Aaftab Munshi Benedict R Gaster Timothy G. Mattson James Fung Dan Ginsburg (2011), OpenCL Programming Guide 1st Edition, Addison-Wesley Professional.

4. clFFT, Library and API documentation, available at: https://github.com/clMathLibraries/clFFT (Accessed 10 October 2020)

5. Improve FFT post-processing performance using clFFT Post-callback (2016), available at https://developer.amd.com/improve-fft-post-processing-performance-using-clfft-post-callback/, (Accessed 10 September 2020).

6. Fuzhou Rockchip Electronics Co., Ltd, “RK3399Pro Datasheet Rev 1.1”, pp 7-18.

7. A.I. Solonina, D.A. Ulahovich, S.M. Arbuzov, E.B. Solovyova (2005), “Osnovi cifrovoy obrabotki signalov” Basics of digital signal processing, 2 Edition, SPB, p 768, pp 589-620.

8. V.S. Priputin, S.Y. Sokolov, N.A. Kandaurov, (2020), “Implementation of the Discrete Frequency Demodulator on Processor with ARM Architecture”, *2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, 19889625.

9. V.R. Magsumov, V.S. Priputin, D.S. Chirov, (2020), “Development and Research of a HF Range Hybrid Filter Bank Based on ARM Processor” *2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO),* 9166064.

РЕАЛИЗАЦИЯ КОСИНУСНО-МОДУЛИРОВАННЫХ ЦИФРОВЫХ ФИЛЬТР БАНКОВ НА БАЗЕ ПРОЦЕССОРА С АРХИТЕКТУРОЙ ARM

Соколов Кирилл Юрьевич, *Московский Технический Университет Связи и Информатики (МТУСИ), Москва, Россия,* ***sokolovkirilluy@gmail.com***
Припутин Владимир Сергеевич, *Московский Технический Университет Связи и Информатики (МТУСИ), Москва, Россия,* ***v.s.priputin@mtuci.ru***
Лобова Елизавета Олеговна, *Московский Технический Университет Связи и Информатики (МТУСИ), Москва, Россия,* ***lizabeth2@mail.ru***

Аннотация

Представлен класс многоканальных косинусно-модулированных банков фильтров (CMFB) анализа, основанных на эффекте модуляции с быстрым дискретно-косинусным преобразованием четвёртого типа (DCT-IV), расчёт которого производится с использованием быстрого преобразования Фурье. В качестве фильтра прототипа использовался низкочастотный фильтр с конечной импульсной характеристикой, сдвинутые по частоте копии которого выполнены с использованием эффективной технологии полифазного представления банка фильтров. Приведено сравнение числа арифметических операций, выполняемых digital down converter (DDC) на базе cascade integral-comb (CIC) и на базе CMFB при различном количестве каналов. Представлено программное описание алгоритма CMFB в виде блок схем с описанием возможностей программных библиотек Opencl и clfft для реализации алгоритма банка фильтров и модуляции DCT-IV на графическом процессоре. Проведено тестирование полученного алгоритма на процессоре семейства ARM и графическом процессоре mali с приведением таблицы с sample rate при разном числе каналов с максимальной загрузкой графического процессора (GPU) и минимальной загрузкой центрального процессора (CPU).

Ключевые слова: *косинусно-модулированный фильтр банк, дискретное косинусное преобразование 4 типа, графический процессор, OpenCL, ARM процессоры, вычислительная сложность*

Литература

1. *Ari Viholainen, Juuso Alhava, and Markku Renfors.* Efficient Implementation of Complex Modulated Filter Banks Using Cosine and Sine Modulated Filter Banks // Hindawi Publishing Corporation EURASIP Journal on Applied Signal Processing Volume (2006), Article ID 58564. С. 1-10.
2. *Henrique S. Malvar.* Signal Processing with Lapped Transforms. Artech House, INC. 1992. С. 67-75.
3. *Aaftab Munshi Benedict R. Gaster Timothy G.* Mattson James Fung Dan Ginsburg. OpenCL Programming Guide 1st Edition, Addison-Wesley Professional, 2011.
4. clFFT, Library and API documentation, available at: https://github.com/clMathLibraries/clFFT (Accessed 10 October 2020).
5. Improve FFT post-processing performance using clFFT Post-callback (2016), available at https://developer.amd.com/improve-fft-post-processing-performance-using-clfft-post-callback/, (Accessed 10 September 2020)
6. Fuzhou Rockchip Electronics Co., Ltd, "RK3399Pro Datasheet Rev 1.1". С. 7-18
7. *Солонина А.И., Улахович Д.А., Арбузов С.М., Соловьева Е.Б.* Основы цифровой обработки сигналов, Изд. 2-у испр. и переаб. СПБ. БЧВ-Петербург, 2005. С. 589-620. 768 с.
8. *Priputin V.S., Sokolov S.Y., Kandaurov N.A.* Implementation of the Discrete Frequency Demodulator on Processor with ARM Architecture // 2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO), 2020, 19889625.
9. *Magsumov V.R., Priputin V.S., Chirov D.S.* Development and Research of a HF Range Hybrid Filter Bank Based on ARM Processor // 2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO), 2020. 9166064.

Информация об авторах:
Соколов Кирилл Юрьевич, *Московский Технический Университет Связи и Информатики (МТУСИ), инженер первой категории НИЛ-4807 НИЧ МТУСИ, Москва, Россия*
Припутин Владимир Сергеевич, *Московский Технический Университет Связи и Информатики (МТУСИ), к.т.н. зав. лаб НИЛ-4807 НИЧ МТУСИ, Москва, Россия*
Лобова Елизавета Олеговна, *Московский Технический Университет Связи и Информатики (МТУСИ), м.н.с. НИЛ-4803 НИЧ МТУСИ, Москва, Россия*